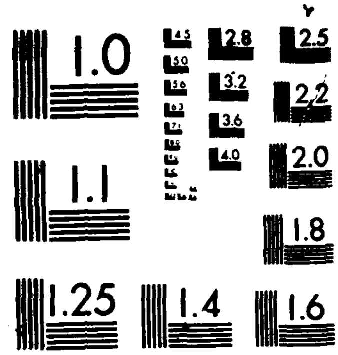
AD-A183 315 ON FINDING THE VERTEX CONNECTIVITY OF GRAPHS(U) 1/1
ILLINOIS UNIV AT URBANA APPLIED COMPUTATION THEORY
GROUP M GIRKAR ET AL MAY 87 ACT-77 N00014-84-C-0149

UNCLASSIFIED F/G 12/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

AD-A183 315

May 1987

UILU-ENG-87-2232
ACT-77

COORDINATED SCIENCE LABORATORY
College of Engineering

2

DTIC FILE COPY

ON FINDING THE VERTEX CONNECTIVITY OF GRAPHS

Milind Girkar
Milind Sohoni

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2232 (ACT-77)	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Joint Services Electronics Program 8b. OFFICE SYMBOL (If applicable) N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-84-C-0149
8c. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: N/A PROJECT NO.: N/A TASK NO.: N/A WORK UNIT ACCESSION NO.: N/A

11. TITLE (Include Security Classification)

On Finding the Vertex Connectivity of Graphs

12. PERSONAL AUTHOR(S)

Girkar, Milind and Sohoni, Milind

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM ____ TO ____	May 1987	5

16. SUPPLEMENTARY NOTATION

N/A

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Vertex connectivity, Graph algorithm, Network flows

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

An implementation of the fastest known algorithm to find the vertex connectivity of graphs with reduced space requirement is presented.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

On finding the vertex connectivity of graphs[1]

Milind Girkar[2]
Milind Sohoni[3]

Abstract

An implementation of the fastest known algorithm to find the vertex connectivity of graphs with reduced space requirement is presented.

1. Introduction

Let $G(V,E)$ be a finite undirected graph with no self-loops and no parallel edges. A set of vertices, S, is called an (a,b) *vertex separator* if $\{a,b\} \subseteq V-S$ and every path connecting a and b passes through at least one vertex of S. Clearly if a and b are connected by an edge, no (a,b) vertex separator exists. We define $N^G(a,b)$ to be $|V|-1$ if $(a,b) \epsilon E$, else it is the least cardinality of an (a,b) vertex separator. The vertex connectivity of G, k_G is defined to be $\min_{a,b \in V} N^G(a,b)$.

When k_G is small, there are well-known linear time algorithms to determine connectivity ($k_G>0$), biconnectivity ($k_G>1$) (see e.g., [4]) and triconnectivity ($k_G>2$) [8,11]. There is an $O(|V|^2)$ algorithm [9] to check four-connectivity ($k_G>3$); others [3,5,7] are of $O(|V||E|)$. For a fixed k, there are some randomised algorithms [1,10] for testing k-connectivity.

In this paper we consider the question of determining k_G, when k_G is large. For this problem, the only known deterministic methods to find it depend on solving maximum flow problems in unit networks [5,7]. (A unit network has the property that the capacity of each edge is one

[1] This work was supported in part by the Joint Services Electronics Program under Grant No. N00014-84-C-0149

[2] Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, IL 61801. The work of this author was supported in part by the National Science Foundation under Grants No. NSF DCR84-10110 and NSF DCR84-06916, the U.S. Department of Energy under Grant No. DOE DE-FG02-85ER25001 and the IBM Donation

[3] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801

and every vertex other than the source or sink has either only one edge emanating from it or one edge entering it.) Of these, the most efficient one is Galil's [7] with a running time of $O(\max(k_G,|V|^{1/2})k_G|E||V|^{1/2})$ with a space requirement of $O((k_G^2+|V|)|E|)$. We improve upon this result by presenting an algorithm that has the same running time as Galil's but with a space requirement of only $O(|V||E|)$.

2. Even's Algorithm

In [3] Even solves the simpler problem (denoted by $P_{G,k}$) of finding whether $k_G \geq k$, for a given G and k. Even's algorithm is as follows:

Let $V=\{v_1,v_2,\cdots,v_n\}$ and let $L_j=\{v_1,v_2,\cdots,v_{j-1}\}$. Define $\hat{G}_j$ to be the graph constructed in the following way. $\hat{G}_j$ contains all the vertices and edges of G; in addition it includes a new vertex s connected by an edge to each vertex in L_j.

(1) For every i and j such that $1 \leq i < j \leq k$, check whether $N^G(v_i,v_j) \geq k$. If for some i and j this test fails then halt; $k_G < k$.

(2) For every j such that $k+1 \leq j \leq |V|$, form $\hat{G}_j$ and check whether $N^{\hat{G}_j}(s,v_j) \geq k$. If for some j this test fails then halt; $k_G < k$.

(3) Halt; $k_G \geq k$.

Whether $N(a,b) \geq k$ can be found out by checking that the value of the maximum flow in the corresponding network is at least k. This involves finding k flow augmenting paths (f.a.p.'s) in the network using the Ford and Fulkerson [6] algorithm. A f.a.p. can be found in $O(|E|)$ time and since k f.a.p.'s need to be found in at most $k^2+|V|$ flow problems, the complexity of Even's algorithm is $O(k^3|E|+k|V||E|)$.

In [7] Galil observes that Even's algorithm can be used to find k_G by progressively solving $P_{G,1}, P_{G,2}, \cdots$ until $P_{G,k+1}$ yields a negative answer; then $k_G = k$. By using Dinic's algorithm [2] to find augmenting paths and modifying Even's algorithm, Galil shows that this can be done in $O(\max(k_G, |V|^{1/2}) k_G |V|^{1/2} |E|)$ using $O((k_G^2 + |V|)|E|)$ memory. Using an approach similar to Galil's we get a reduced space bound.

3. The Algorithm

First we simplify Even's algorithm as follows:

In the first step instead of checking whether $N^G(v_i, v_j) \geq k$, we do some additional work and find $N^G(v_i, v_j)$ and then trivially check whether this is greater than or equal to k. It will turn out that the extra work will not change the time complexity of the algorithm.

The outline of the algorithm is as follows.

(1) Initialize k to 1, MIN to $|V|-1$.

(2) For every i such that $1 \leq i \leq k-1$, find $N^G(v_i, v_k)$.

(3) Use the results of step 2 to update MIN to $\min(\min_{1 \leq i \leq k-1} N^G(v_i, v_k), MIN)$

(4) If $MIN < k$ then halt; $k_G = MIN$.

(5) For every j such that $k+1 \leq j \leq |V|$, check whether $N^{\tilde{G}_j}(s, v_j) \geq k$. If this test fails for any j, then halt; $k_G = k-1$.

(6) Increment k by one, go to step 2.

The correctness of the above algorithm follows from the results in Even [3]. We now analyze the time and space requirement of the algorithm. We store the graphs $\tilde{G}_j$ $(2 \leq j \leq |V|)$ along with the current flow values in the corresponding networks. In each iteration the computationally intensive steps are clearly 2 and 5. In the k^{th} iteration, we solve $k-1$ maximum flow

problems in step 2 and using the flow values computed in the $k-1^{th}$ iteration for the networks corresponding to $\tilde{G}_j$, we check whether $N^{\tilde{G}_j}(s,v_j)\geq k$ in step 5 by finding at most one f.a.p. in each of the corresponding networks. Using Dinic's algorithm [2] step 2 can be done in $O(k|E||V|^{1/2})$ time and step 5 in $O(|V||E|)$ time since an f.a.p. can be found in linear time. Thus the running time of the algorithm is $O(k_G^2|E||V|^{1/2}+k_G|V||E|) = O((k_G+|V|^{1/2})k_G|E||V|^{1/2}) = O(\max(k_G,|V|^{1/2})k_G|E||V|^{1/2})$ which is the same as Galil's algorithm. However, the space requirement is only $O(|V||E|)$ because the flow values for at most $|V|$ maximum flow problems have to be stored and each requires $O(|E|)$ space.

Acknowledgment. The authors wish to thank V. Ramachandran for introducing us to the problem in a course on graph algorithms and for her many helpful suggestions during the preparation of this report.

REFERENCES

1. Becker, M. et. al. *A probabilistic algorithm for vertex connnectivity of graphs.* **Inform. Proc. Lett.** (1982) vol. 15, pp. 135–136.

2. Dinic, E. A. *Algorithm for solution of a problem of maximum flow in a network with power estimation.* **Soviet Math. Dokl.** (1970) vol. 11, pp. 1277–1280.

3. Even, S. *An algorithm for determining whether the connectivity of a graph is at least k.* **SIAM Journal of Computing** (1975) vol. 4, pp. 393–396.

4. ——. **Graph Algorithms**. Computer Science Press, Rockville, MD, 1979.

5. Even, S. and R. E. Tarjan. *Network flow and testing graph connectivity.* **SIAM Journal of Compu ing** (1975) vol. 4, pp. 507–518.

6. Ford, L. R. and D. R. Fulkerson. **Flows in Networks**. Princeton Press, Princeton, NJ, 1962.

7. Galil, Z. *Finding the vertex connectivity of graphs.* SIAM Journal of Computing (February 1980) vol. 9, pp. 197–199.

8. Hopcroft, J. E. and R. E. Tarjan. *Dividing a graph into triconnected components.* SIAM Journal of Computing (1973) pp. 135–158.

9. Kanevsky, A. and V. Ramachandran. "Improved algorithms for graph four-connectivity", Working Paper 87–14, Coordinated Science Labaratory, University of Illinois at Urbana–Champaign, Urbana, IL, 1987.

10. Linial, N., L. Lovasz and A. Wigderson. *A physical interpretation of graph connectivity, and its algorithmic applications.* Proc. 28th IEEE Ann. Symp. on Foundations of Comp. Sci. (1985) pp. 464–467.

11. Miller, G. L. and V. Ramachandran. *A new graph triconnectivity algorithm and its parallelization.* Proc. of the 19th Annual ACM Symposium on Theory of Computing (May 1987).

END

9 — 87

DTIC